

# Objects as context for part detection

Abel Gonzalez-Garcia

a.gonzalez-garcia@sms.ed.ac.uk

Davide Modolo

d.modolo@sms.ed.ac.uk

Vittorio Ferrari

vferrari@staffmail.ed.ac.uk

University of Edinburgh

## Abstract

We present a semantic part detection approach that effectively leverages object information. We use the object appearance and its class as indicators of what parts to expect. We also model the expected relative location of parts inside the objects based on their appearance. We achieve this with a new network module, called OffsetNet, that efficiently predicts a variable number of part locations within a given object. Our model incorporates all these cues to detect parts in the context of their objects. This leads to significantly higher performance for the challenging task of part detection compared to using part appearance alone (+5 mAP on the PASCAL-Part dataset). We also compare to other part detection methods on both PASCAL-Part and CUB200-2011 datasets.

## 1. Introduction

Semantic parts play an important role in visual recognition. They offer many advantages such as lower intra-class variability than whole objects, higher robustness to pose variation, and their configuration provides useful information about the aspect of the object. For these reasons, part-based models have gained attention for tasks such as fine-grained recognition [1–4], object class detection and segmentation [5, 6], articulated pose estimation [7–10], and attribute prediction [11–13]. Moreover, part localizations deliver a more comprehensive image understanding, enabling reasoning about object-part interactions in semantic terms. Nonetheless, many part-based models detect each part based only on their local appearance, using simple techniques that were originally designed for object detection [4, 5, 13]. Furthermore, some other works use even simpler approaches relying on the assumption that convolutional filters can act as part detectors [11, 14, 15]. Here we take part detection one step further and provide a specialized approach that exploits the unique nature of this task.

Parts are highly dependent on the objects that contain them. Hence, objects provide valuable cues to help detecting parts, creating an advantage over detecting them independently. First, the class of the object gives a firm indication of what parts should be inside it, i.e. only those belonging to that object class. For example, a dark round patch should be more confidently classified as a wheel if it is on

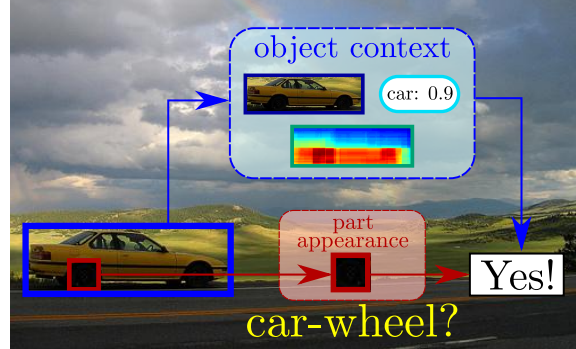


Figure 1. **Motivation for our model.** Part appearance alone might not be sufficiently discriminative in some cases. Our model uses object context to resolve ambiguities and help part detection.

a car, rather than on a dog (fig. 1). Furthermore, by looking at the object appearance we can determine in greater detail which parts might be present. For example, a profile view of a car suggests the presence of a car door, and the absence of the licence plate. This information comes mostly through the viewpoint of the object, but also from other factors, such as the type of object (e.g. van), or whether the object is truncated (e.g. no wheels if the lower half is missing). Second, objects also provide information about the location and shape of the parts they contain. Semantic parts appear in very distinctive locations within objects, especially given the object appearance. Moreover, they appear in characteristic relative sizes and aspect ratios. For example, wheels tend to be near the lower corners of car profile views, often in a square aspect ratio, and appear rather small.

In this work, we propose a dedicated part detection model that leverages all of the above object information. We start from a popular Convolutional Neural Network (CNN) detection model [16], which considers the appearance of local image regions only. We extend this model to incorporate object information that complements part appearance by providing context in terms of object appearance, class and the relative locations of parts within the object.

We evaluate our part detection model on all 16 object classes in the PASCAL-Part dataset [5]. We demonstrate that adding object information is greatly beneficial for the difficult task of part detection, leading to considerable performance improvements. Compared to a baseline detection model that considers only the local appearance of parts, our

model achieves a +5 mAP improvement. We also compare to methods that report part localization in terms of bounding-boxes [1–3, 5, 11] on PASCAL-Part and CUB200-2011 [17]. We outperform [1, 2, 5, 11] and match the performance of [3]. We achieve this by an effective combination of the different object cues considered, demonstrating their complementarity. Moreover our approach is general as it works for a wide range of object classes: we demonstrate it on 16 classes, as opposed to 1–7 in [1–11, 13–15, 18–23] (only animals and person). Finally, we perform fully automatic object and part detection, without using ground-truth object locations at test time [2, 3, 5, 6].

## 2. Related work

**Part-based models.** Part-based models have been used in many tasks such as object detection [5, 24–26] and segmentation [27], fine-grained recognition [3, 4, 13, 14, 28, 29], human pose-estimation [7–10, 30], head and face detection [31, 32], attribute prediction [12, 13, 18], action classification [11] and scene classification [33]. These methods can be divided into two groups, depending on their definition of ‘part’. The first group defines a part as any image patch discriminative for the object class [14, 18, 24–27, 33]. The second group defines parts semantically (e.g. ‘saddle’) [3–5, 7–10, 12, 13, 27–32, 34]. Our work belongs to the second: we propose a method to improve semantic part detection by exploiting the context provided by their objects.

**CNN-based semantic part-based models.** In recent years, CNN-based representations are quickly replacing hand-crafted features [35, 36] in many domains, including semantic part-based models [1, 6, 11, 19–21, 27, 29, 30, 37]. Our work is related to those that explicitly train CNN models to localize semantic parts using bounding-boxes [1, 11], as opposed to keypoints [14, 29] or segmentation masks [6, 19–21, 27, 31]. Many of these works [11, 14, 29, 31] detect the parts used in their models based only on local part appearance, independently of their objects. Moreover, they use parts as a means for object or action and attribute recognition, they are not interested in part detection itself.

**Using context for recognition.** Several works exploit global image context to help detecting objects inside it [32, 38–44]. In analogy, we exploit object context to help detecting parts inside them.

Several fine-grained recognition works [3, 22, 23] use nearest-neighbors to transfer part location annotations from training objects to test objects. They do not perform object detection, as ground-truth object bounding-boxes are used at both training and test time. Here, instead, at test time we jointly detect objects and their semantic parts.

Both [1, 6] use object information to refine part detections as a post-processing step. Part-based R-CNN [1] refines R-CNN [45] part detections by using nearest-neighbors from training samples. Our model, instead in-

tegrates object information also within the network, which allows us to deal with several object classes simultaneously, as opposed to only one [1]. Additionally, we refine part detections with a new network module, Offset Net, which is more accurate and efficient than nearest-neighbors. The method of Wang et al. [6] is demonstrated only on 5 very similar classes from PASCAL-Part [5] (all quadrupeds), and on fully visible object instances from a manually selected subset of the test set (10% of the full test set). Instead, we show results on 105 parts over all 16 classes of PASCAL-Part, using the entire dataset. Moreover, [6] uses manually defined object locations at test time, whereas we detect both objects and their parts fully automatically at test time.

## 3. Method

We define a new detection model specialized for parts which takes into account the context provided by the objects that contain them. This is the key advantage of our model over traditional part detection approaches, which detect parts based on their local appearance alone, independently of the objects [11, 14, 29, 31]. We build on top of a baseline part detection model (sec. 3.1) and include various cues based on object class (sec. 3.2.2), object appearance (sec. 3.2.3), and the relative location of parts on the object (sec. 3.3). Finally, we combine all these cues to achieve more accurate part detections (sec. 3.4).

**Model overview.** Fig. 2 gives an overview of our model. First, we process the input image through a series of convolutional layers. Then, the Region of Interest (RoI) pooling layer produces feature representations from two different kind of region proposals, one for parts (red) and one for objects (blue). Each part region gets associated with a particular object region that contains it (sec. 3.2.1). Features for part regions are passed on to the part appearance branch, which contains two Fully Connected (FC) layers (sec. 3.1). Features for object regions are sent to both the object class (sec. 3.2.2) and object appearance (sec. 3.2.3) branches, with three and two FC layers, respectively.

For each part proposal, we concatenate the output of the part appearance branch with the outputs of the two object branches for its associated object proposal. We pass this refined part representation (purple) on to a part classification layer and a bounding-box regression layer (sec. 3.4).

Simultaneously, the relative location branch (green) also produces classification scores for each part region based on its relative location within the object (sec. 3.3). We combine the above part classification scores with those produced by relative location (big + symbol, sec. 3.4), obtaining the final part classification scores. The model outputs these and regressed bounding-boxes.

### 3.1. Baseline model: part appearance only

As baseline model we use the popular Fast R-CNN [16], which was originally designed for object detection. It is

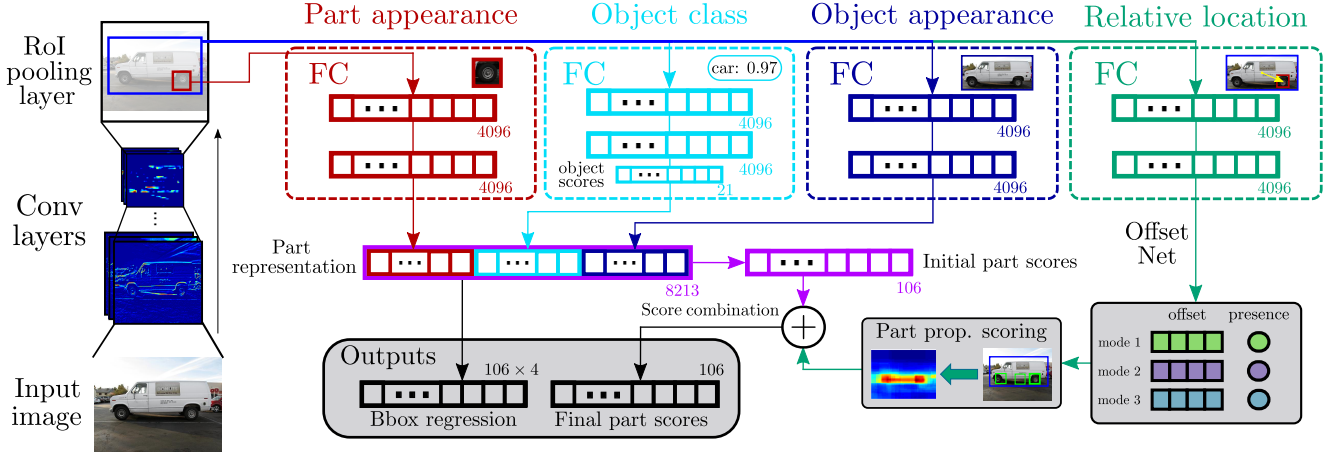


Figure 2. **Overview of our part detection model.** The model operates on part and object region proposals, passing them through several branches, and outputs part classification scores and regressed bounding-boxes. This example depicts the relative location branch for only object class car. In practice, however, it processes all object classes simultaneously. When not explicitly shown, a small number next to the layer indicates its dimension for the PASCAL-Part [5] case, with a total of 20 object classes and 105 parts.

based on a CNN that scores a set of region proposals [46] by processing them through several layers of different types. The first layers are convolutional and they process the whole image once. Then, the ROI pooling layer extracts features for each region proposal, which are later processed by several FC layers. The model ends with two sibling output layers, one for classifying each proposal into a part class, and one for bounding-box regression, which refines the proposal shape to match the extent of the part more precisely. The model is trained using a multi-task loss which combines these two objectives. This baseline corresponds to the part appearance branch in fig. 2.

We follow the usual approach [16] of fine-tuning for the used dataset on the current task, part detection, starting from a network pre-trained for image classification [47]. The classification layer of our baseline model has as many outputs as part classes, plus one output for a generic background class. Note how we have a single network for all part classes in the dataset, spanning across all object classes.

### 3.2. Adding object appearance and class

The baseline model tries to recognize parts based only on the appearance of individual region proposals. In our first extension, we include object appearance and class information by integrating it inside the network. We can see this as selecting an adequate contextual spatial support for the classification of each proposal into a part class.

#### 3.2.1 Supporting proposal selection

Our models use two types of region proposals (sec. 4). *Part proposals* are candidate regions that might cover parts. Analogously, *object proposals* are candidates to cover objects. The baseline model uses only part proposals. In our models, instead, each part proposal  $p$  is accompanied by a *supporting object proposal*  $S_{sup}(p)$ , which must fulfill two requirements (fig. 3). First, it needs to contain the part proposal, i.e. at least 90% of  $p$  must be inside  $S_{sup}(p)$ . Sec-

ond, it should tightly cover the object that contains the part, if any. For example, if the part proposal is on a wheel, the supporting proposal should be on the car that contains that wheel. To achieve this, we select the highest scored proposal among all object proposals containing  $p$ , where the score is the object classification score for any object class.

Formally, let  $p$  be a part proposal and  $S(p)$  the set of object proposals that contain  $p$ . Let  $\phi_{obj}^k(S_n)$  be the classification score of proposal  $S_n \in S(p)$  for object class  $k$ . These scores are obtained by first passing all object proposals through three FC layers as in the object detector [16]. We select the supporting proposal  $S_{sup}(p)$  for  $p$  as

$$S_{sup}(p) = \underset{S_n \in S(p)}{\operatorname{argmax}} \left[ \max_{k \in \{1, \dots, K\}} \phi_{obj}^k(S_n) \right], \quad (1)$$

where  $K$  is the total number of object classes in the dataset.

#### 3.2.2 Object class

The class of the object provides cues about what part classes might be inside it. For example, a part proposal on a dark round patch cannot be confidently classified as a wheel based solely on its appearance (fig. 1). If the corresponding supporting object proposal is a car, the evidence towards it being a wheel grows considerably. On the other hand, if the supporting proposal is a dog, the patch should be confidently classified as *not* a wheel.

Concretely, we process convolutional features pooled from the supporting object proposal through three FC layers (fig. 2). The third layer performs object classification and outputs scores for each object class, including a generic background class. These scores can be seen as object semantic features, which complement part appearance.

#### 3.2.3 Object appearance

The appearance of the object might bring even more detailed information about what part classes it might contain. For example, the side view of a car indicates that we can expect to find wheels, but not a licence plate. We model



Figure 3. **Examples of supporting proposal selection.** For each part proposal (yellow), we select as its supporting proposal (blue) the highest scored among the object proposals that contain it.

object appearance by processing the convolutional features of the supporting proposal through two FC connected layers (fig. 2). These type of features have been shown to successfully capture the appearance of objects [48, 49].

### 3.3. Adding relative location

In this section, we add another type of information that could be highly beneficial: the relative location of the part with respect to the object. Parts appear in very distinct and characteristic relative locations and sizes within the objects. Fig. 4a shows examples of prior relative location distributions for some part classes as heatmaps. These are produced by accumulating all part ground-truth bounding-boxes from the training set, in the normalized coordinate frame of the bounding-box of their object. Moreover, this part location distribution can be sharper if we condition it on the object appearance, especially its viewpoint. For example, the *car-wheel* distribution on profile views of cars will only have two modes (fig. 4b) instead of the three shown in fig. 4a.

We incorporate this cue into our model by scoring each part proposal using its relative location with respect to the object. This indicates the probability that a proposal belongs to a certain part class, based purely on its location (it does not depend on part appearance).

Our relative location model is specific to each part class within each object class (e.g. a model for car-wheel, another for cat-tail). Below we explain the model for one particular object and part class. Given a proposal  $o$  of that object class, our model suggests a set of windows  $\Lambda(o) = \{w_i\}_{i=1}^I$  inside  $o$  where instances of that part class are likely to be. Naturally, these windows will also depend on the appearance of  $o$ . For example, given a car profile view, our model suggests square windows on the lower corners as likely to contain wheels (fig. 4b top). Instead, an oblique view of a car will also suggest wheels towards the lower central region, as well as a more elongated aspect ratio for the wheels on the side (fig. 4b bottom).

Let  $O$  be a set of object detections for a particular image, i.e. object proposals with high score after being processed through non-maxima suppression [25]. We produce them automatically using standard Fast R-CNN [16]. Let  $\phi_{obj}(o)$  be the score of detection  $o \in O$  for the considered object class. We compute the relative location score  $\phi_{rl}(p)$  for part proposal  $p$  using its overlap with all suggested windows

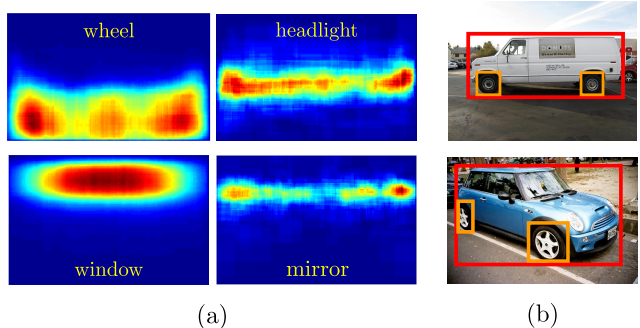


Figure 4. **Prior distributions and examples of part relative locations.** (a) Heatmaps created using part ground-truth bounding-boxes normalized to the object bounding-box, using all car training samples. (b) Examples of part ground-truth bounding-boxes inside object bounding-boxes of class car, in different viewpoints.

$$\phi_{rl}(p) = \max_{w_i \in \Lambda(o), o \in O} (\text{IoU}(p, w_i) \cdot \xi_i \cdot \phi_{obj}(o)), \quad (2)$$

where we use Intersection-over-Union (IoU) to measure overlap. Here,  $\xi_i$  is a confidence weight associated to each individual window  $w_i$ , and  $\phi_{obj}(o)$  weights how much all windows  $\Lambda(o)$  suggested by detection  $o$  should be trusted. Consequently, detections with higher score provide stronger cues through more suggested windows. Fig. 5 shows an example, where we show only one *car* detection for clarity.

Below we explain two alternative ways to generate the suggested windows  $w_i$  and their confidence  $\xi_i$ . The first uses nearest-neighbors (sec. 3.3.1), whereas the second uses a new CNN architecture we dub *Offset Net* (sec. 3.3.2).

#### 3.3.1 Nearest-neighbors (NN)

Parts of similar-looking objects tend to occupy similar relative locations within them. We exploit this phenomenon: given an object detection  $o$  in a test image, we retrieve part ground-truth bounding-boxes of objects similar to it in the training set and warp them into the coordinate frame of  $o$ . Ideally, these suggested windows will cover part instances on the test object.

**Training database.** For each object and part class combination, we create a database of pairs  $\{(\psi(o_n), \Delta v_n)\}_{n=1}^N$  from the training set. Here,  $\psi(o_n)$  represents the appearance of object proposal  $o_n$ . Concretely, we use L2-normalized CNN features. Some works [48, 49] have used FC features to represent appearance for clustering purposes, as they are more robust to pose than convolutional features. However, we prefer using features from the last convolutional layer, as we are interested in preserving the internal spatial structure of the object. The second element  $\Delta v_n$  is an offset: a 4D vector that maps object proposal  $o_n$  to a part ground-truth bounding-box inside it. If  $o_n$  contains multiple instances of the part inside (e.g. two wheels in a car),  $\Delta v_n$  represents a set of vectors instead, one per part instance.

**Testing.** At test time, given a query object detection  $o$ , our method retrieves its  $L$  nearest-neighbors from the training database and their set of offset vectors  $\{\Delta v_l\}_{l=1}^L$ . Note that the total number of vectors might be greater than  $L$ , as one



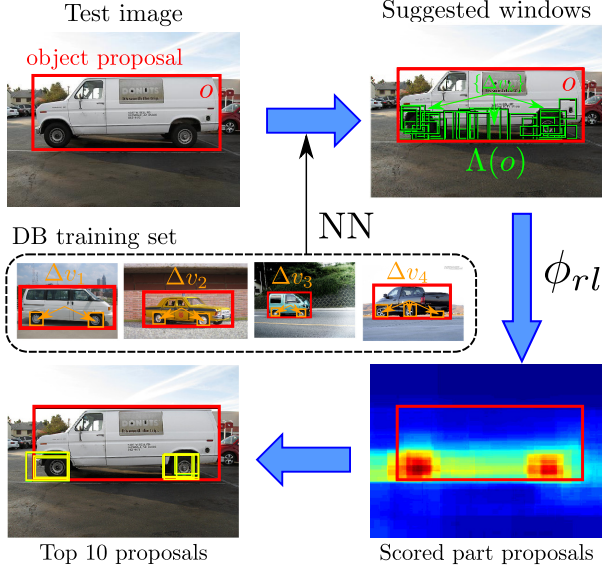


Figure 5. **Example of scoring part proposals based on relative location.** Part class: car-wheel. Each object detection suggests windows likely to contain car-wheel within it. We score part proposals by computing their max IoU with any suggested window, and weighting them by confidence and object detection score. The top scored proposals, shown here as an example, nicely cover part instances. The suggested windows can be provided by nearest-neighbors (sec. 3.3.1), as in here, or by Offset Net (sec. 3.3.2).

$\Delta v_l$  could comprise several vectors. We then obtain the set of suggested windows at test time by applying the retrieved vectors to  $o$ :  $\Lambda(o) = \{o + \Delta v_l\}_{l=1}^L$ . When  $|\Delta v_l| > 1$ , the offset operator  $+$  is applied multiple times to  $o$ , resulting in  $|\Delta v_l|$  windows. The suggested windows  $\Lambda(o)$  cover likely locations for the part class, given the appearance of object  $o$  in the test image (fig. 5). In the nearest-neighbor case, all confidence weights  $\xi$  in (2) are set to 1.

### 3.3.2 Offset Net

Using nearest-neighbors has two main drawbacks. First, it is cumbersome since we need to store the database of training objects and part bounding-boxes. Second, it is slow since at test time we have to compute many distances to retrieve the nearest-neighbors of detection  $o$ .

We propose here a more elegant and faster approach: generate suggested windows  $\Lambda(o)$  using a special kind of CNN, which we dub *Offset Net* (see fig. 2, Relative location branch). Offset Net directly learns to regress from the appearance of  $o$  to the relative location of a part class within it (i.e. learns to produce the offset that needs to be applied to  $o$  to generate  $\Lambda(o)$ ). Nearest-neighbor instead mimics this effect by copying offset vectors from similar objects in the training set. Intuitively, a CNN is a good framework to learn this regressor, as the activation maps of the network contain localized information about the parts of the object [50, 51].

**Model.** Formally, the input to Offset Net is the test image and the set of object detections  $O$ . For each detection  $o \in O$ , the network outputs a set of 4D offset vectors  $\Delta v$  for

each part class. It can contain multiple vectors as some objects have multiple instances of the same part in them (e.g. cars with multiple wheels).

Offset Net generates each offset vector in  $\Delta v$  through a regression layer. The nearest-neighbor handled the case  $|\Delta v| > 1$  automatically by storing multiple vectors per object instance of the training set. To enable Offset Net to output multiple vectors we build multiple parallel regression layers. We set the number of parallel layers to the number of modes of the prior distribution for each part class (fig. 4). For example, the prior *car-wheel* has three modes, leading to three offset regression layers in Offset Net (fig. 2). On the other hand, Offset Net only has one regression layer for *person-head*, as its prior distribution is unimodal.

In some cases, however, not all modes are active for a particular object instance (e.g. profile views of cars only have two active modes out of the three, fig. 4b). For this reason, each regression layer in Offset Net has a sibling layer that predicts the presence of that mode in the input detection  $o$ , and outputs a presence score  $\rho$ . This way, even if the network outputs multiple offset vectors, only those with a high presence score will be taken into account. This construction effectively enables Offset Net to produce a variable number of output offset vectors, depending on the input  $o$ .

**Training.** We train the offset regression layers using a smooth-L1 loss, as in the bounding-box regression of Fast R-CNN [16]. We train the presence score layer using a logistic log loss:  $L(x, c) = \log(1 + e^{-cx})$ , where  $x$  is the score produced by the network, and  $c$  is a binary label indicating whether the current mode is present ( $c = +1$ ) or not ( $c = -1$ ). We generate  $c$  using annotated ground-truth bounding-boxes (sec. 4). This loss implicitly normalizes score  $x$  using the sigmoid function. After training, we add a sigmoid layer to explicitly normalize the output presence score:  $\rho = 1/(1 + e^{-x}) \in [0, 1]$ .

**Testing.** At test time, given an input detection  $o$ , Offset Net generates  $M$  pairs  $\{(\delta v_i, \rho_i)\}_{i=1}^M$  of offset vectors  $\delta v_i \in \Delta v$  and presence scores  $\rho_i$  for each part class, where  $M$  is the number of modes in the prior distribution. We apply the offset vectors  $\delta v_i$  to  $o$ , producing a set of suggested windows  $\Lambda(o) = \{o + \delta v_i\}_{i=1}^M = \{w_i\}_{i=1}^M$ . Therefore, suggested windows more likely to be present have a higher confidence. We then compute the relative location score  $\phi_{rl}(p)$  with eq. (2), using the presence scores  $\rho_i$  as confidence weights:  $\xi_i = \rho_i$ .

Fig. 6 shows examples of windows suggested by Offset Net, along with their presence score and a heatmap generated by scoring part proposals using eq. (2). We can see how the suggested windows cover very likely areas for part instances on the input objects, and how the presence scores are crucial to decide which windows should be relied on.

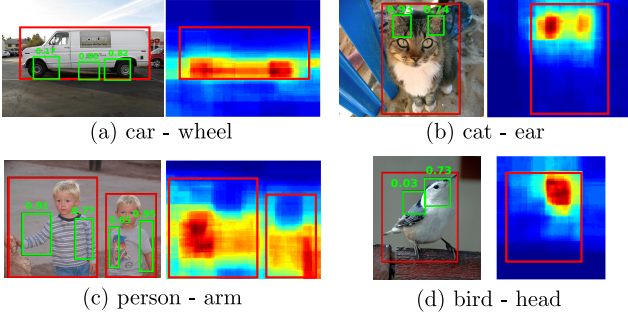


Figure 6. **Offset Net results.** Examples of windows suggested by Offset Net (green) for different part classes, given the input object detections (red). We also show the heatmap generated by scoring all part proposals, showing how highly scored proposals occupy areas likely to contain part instances. The presence scores clearly indicate which suggested windows should be relied on (e.g. the second head of the bird and the middle wheel of the van have very low scores and are discounted in  $\phi_{r,l}$ ).

### 3.4. Cue combination

We have presented multiple cues that can help part detection. These cues are complementary, so our model needs to effectively combine them.

We concatenate the output of the part appearance, object class and object appearance branches and pass them on to a part classification layer that combines them and produces initial part classification scores (purple in fig. 2). Therefore, we effectively integrate object context into the network, resulting in the automatic learning of object-aware part representations. We argue that this type of context integration has greater potential than just a post-processing step [1, 6].

The relative location branch, however, is special as its features have a different nature and much lower dimensionality (4 vs 4096). To facilitate learning, instead of directly concatenating them, this branch operates independently of the others and computes its own part scores. Therefore, we linearly combine the initial part classification scores with those delivered by the relative location branch (big + in fig. 2). For some part classes, the relative location might not be very indicative due to high variance in the training samples (e.g. *cat-nose*). In some other cases, relative location can be a great cue (e.g. the position of *cow-torso* is very stable across all its instances). For this reason, we learn a separate linear combination for each part class. We do this by maximizing part detection performance on the training set, using grid search on the mixing weight in the  $[0, 1]$  range. We define the measure of performance in sec. 5.

## 4. Implementation details

**Proposals.** Object proposals [46, 52, 53] are designed to cover whole objects, and might fail to acknowledge separations between parts lacking changes in texture or color, such as the wing and the torso of a bird. To alleviate this issue, we changed the standard settings of Selective Search [46], by decreasing the minimum box size to 10. This results in ad-

equating proposals even for parts: reaching 71.4% recall with around 3000 proposals ( $\text{IoU} > 0.5$ ). For objects, we keep the standard Selective Search settings (minimum box size 20), resulting in around 2000 proposals.

**Training the part detection network.** Our networks are pre-trained for image classification on ILSVRC12 [54] and fine-tuned on PASCAL-Part [5] for part detection, or on PASCAL VOC 2010 [55] for object detection, using Mat-ConvNet [56]. Fine-tuning for object detection follows the Fast R-CNN procedure [16]. For the part detection fine-tuning we changed the following settings. Positive samples for parts overlap any part ground-truth  $> 0.6$  IoU, whereas negative samples overlap  $< 0.3$ . We trained for 16 epochs, with learning rate  $10^{-3}$  for the first 12 and  $10^{-4}$  for the remaining 4. The object detection layers used in the object class branch and in the supporting proposal selection are trained for the standard Fast R-CNN object detection loss. The layers in the object appearance branch are jointly trained with the part detection network to provide a tailored object appearance representation for the part detection task.

**Training Offset Net.** We need object samples and part samples to train Offset Net. Our object samples are all object ground-truth bounding-boxes and object proposals with  $\text{IoU} \geq 0.7$  in the training set. Our part samples are only part ground-truth bounding-boxes. We split the horizontal axis in  $M$  regions, where  $M$  is the number of modes in the part class prior relative location distribution. We assign each part ground-truth bounding-box in the object to the closest mode. If a mode has more than one part bounding-box assigned, we pick one at random. All layers except the top ones are initialized with a Fast R-CNN network trained for object detection. Similarly to the other networks, we train it for 16 epochs, but with learning rates  $10^{-4}$  and  $10^{-5}$ . For clarity, in fig. 2 we use the same convolutional features for all branches. In practice, Offset Net uses its own convolutional layers, as they also are fine-tuned for its task.

Upon acceptance, we will release our code and models.

## 5. Results

### 5.1. Validation of our model

**Dataset.** We present results on PASCAL-Part [5], which has pixel-wise part annotations for the images of PASCAL VOC 2010 [55]. For our experiments we fit a bounding-box to each part segmentation mask. We pre-process the set of part classes as follows. We discard additional information on semantic part annotations, such as ‘front’ or ‘left’ (e.g. both “car wheel front left” and “car wheel back right” become *car-wheel*). We merge continuous subdivisions of the same semantic part (“horse lower leg” and “horse upper leg” become *horse-leg*). Finally, we discard tiny parts, with average width and height over the training set  $\leq 15$  pixels (e.g. “bird eye”), and rare parts that appear  $< 10$  times (e.g. “bicycle headlight”). After this pre-processing, we obtain

Model	Obj. App	Obj. Cls	Rel Loc	mAP
Baseline [16]				22.1
Obj. appearance	✓			25.1
Obj. class		✓		23.0
Obj. app + cls	✓	✓		25.7
All-NN (no app)			✓	23.5
5-NN			✓	23.9
Offset Net ( $M = 1$ )			✓	24.3
Offset Net			✓	24.7
Obj. app + 5-NN	✓		✓	26.2
Obj. app + Offset Net	✓		✓	26.8
Full (Obj. app + cls + Offset Net)	✓	✓	✓	27.4
Baseline [16] (bbox-reg)				24.5
Full (bbox-reg)	✓	✓	✓	29.5
Baseline [16] (VGG16, bbox-reg)				35.8
Full (VGG16, bbox-reg)	✓	✓	✓	40.1

Table 1. **Part detection results on PASCAL-Part.** All rows except the last two use AlexNet. The baseline model uses only part appearance. All other models include it too.

a total of 105 part classes for 16 object classes. We train our methods on the `train` set and test them on the `val` set (the `test` set is not annotated in PASCAL-Part). We note how we are the first work to present fully automatic part detection results on the whole PASCAL-Part dataset.

**Performance measure.** Just before measuring performance we remove duplicate detections using non-maxima suppression [25]. We measure part detection performance using Average Precision (AP), following the PASCAL VOC protocol [55]. We consider a part detection to be correct if its IoU with a ground-truth part bounding-box is  $> 0.5$ .

**Baseline results.** As base network in all models we use AlexNet [47], unless stated otherwise (convolutional layers on the leftmost column of fig. 2). Tab. 1 presents part detection results. The baseline model achieves only 22.1 mAP without bounding-box regression (sec. 3.1). As a reference, the same model, when trained and evaluated for object class detection, achieves 48.5 mAP on PASCAL VOC 2010 [55], which contains the same object classes as PASCAL-Part. This massive difference in performance demonstrates the inherent difficulty of the part detection task.

**Adding object appearance and class.** By adding object appearance (sec. 3.2.3), performance increases by +3 mAP, which is a significant improvement. Adding object class (sec. 3.2.2) also helps, albeit less so (+0.9 mAP). This indicates that the appearance of the object contains extra knowledge relevant for part discrimination (e.g. viewpoint), which the object class alone cannot provide. Furthermore, the combination of both types gives a small additional boost (+0.6 mAP compared to using only object appearance). Although in principle object appearance subsumes its class, having a more explicit and concise characterization of the class is beneficial for part discrimination.

**Adding relative location.** Our relative location models (sec. 3.3) also bring improvements. To better understand the effects of our design choices, we evaluate various ablated versions of our models. We start with *All-NN*, a version of our nearest-neighbor approach (sec. 3.3.1) that uses

all training instances, instead of only the ones most similar to the test object. This is essentially a location prior, independent of the object appearance (fig. 4a). All-NN already brings +1.4 mAP improvement, showing that location priors can indeed be helpful. Now we set  $L = 5$ , effectively conditioning on object appearance. The 5-NN model brings a larger improvement (+1.8 mAP), demonstrating the role of object appearance in modulating the relative location cue.

We present results for two versions of our Offset Net model (sec. 3.3.2). In the first one, we fix the number of modes  $M$  to just 1 for all part classes, regardless of the complexity of the prior distribution. In this simplistic setting, Offset Net already surpasses the 5-NN approach, and outperforms the baseline by +2.2 mAP. This indicates that even with only 1 suggested window, Offset Net provides a strong relative location cue. When setting  $M$  based on the prior distribution as explained in sec. 3.3.2, the improvement further rises to +2.6 mAP.

Both our relative location models capture the spirit of this work and help part discrimination. Given Offset Net offers better performance, higher computational efficiency and lower memory footprint than nearest-neighbors, it is our method of choice for our final model.

**Combining cues.** Finally, we now combine all our cues as in sec. 3.4 (always using also part appearance). First, we combine each of our relative location models with object appearance. Both combinations are beneficial and surpass each cue alone. In this setting, Offset Net still brings higher improvements than 5-NN.

Our best model (full) combines all cues and achieves +5.3 mAP over the baseline. We regard this as a substantial improvement, especially considering the high difficulty of the task, as demonstrated by the rather low baseline performance. This result shows that all cues we propose are indeed complementary to part appearance and to each other; when combined, all contribute to the final performance.

We also tested the baseline and our model using bounding-box regression (bbox-reg in tab. 1). This is beneficial in all cases. Importantly, the improvement brought by our full model over the baseline (+5 mAP) is similar to the one without bounding-box regression (+5.3 mAP).

**Example detections.** Fig. 7 shows some part detection examples for both the baseline and our full model (without bounding-box regression). In general, our model localizes parts more accurately, fitting the part extent more tightly (fig. 7a, 7e). Moreover, it also finds some part instances missed by the baseline (fig. 7b, 7c). Our method uses object detections automatically produced by Fast R-CNN [16]. When these are inaccurate, our model can sometimes produce worse part detections than the baseline (fig. 7f).

**Runtime.** We report runtimes on a Titan X GPU. The baseline takes 4.3s/im, our model 7.1s/im. Note how we also output object detections, which the baseline does not.



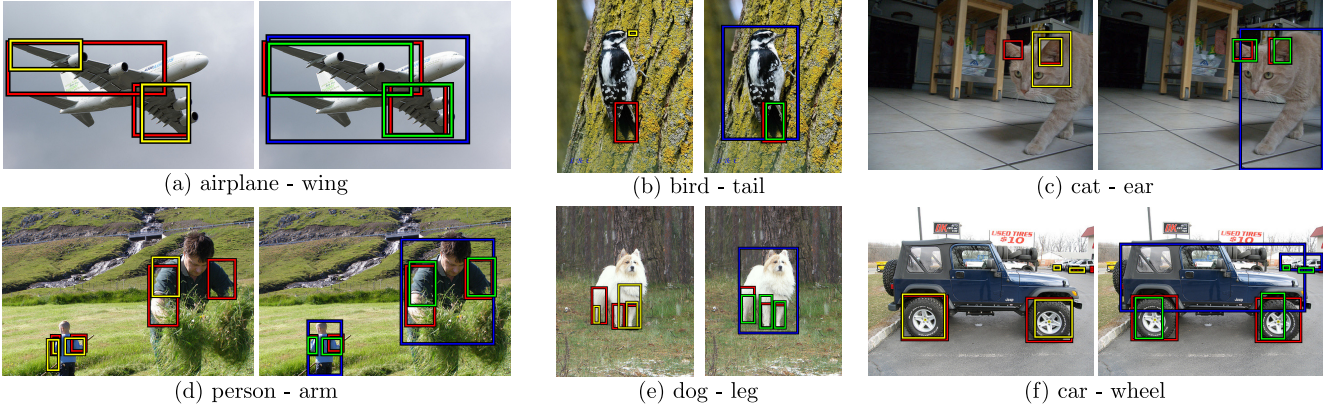


Figure 7. **Qualitative results.** Example part detections for the baseline model (yellow) and our model combining all cues (green). We also show part ground-truth bounding-boxes (red), and object detections output by our method (blue).

**Results for VGG16.** We present results for the deeper VGG16 network [57]. The relative performance of our model and the baseline is analog to the AlexNet case, but with much higher mAP values. The baseline achieves 35.8 mAP with bounding-box regression. Our full model achieves 40.1 mAP, which is a substantial improvement of magnitude comparable to the AlexNet case (+4.3 mAP).

## 5.2. Comparison to other methods

We compare here our full (bbox-reg) model (tab. 1) to several prior works on detecting parts up to a bounding-box [1, 5, 11]. We use AlexNet, which is equivalent to the networks used in [1, 3, 11]. Tab. 2 summarizes all results.

**Chen et al. [5].** We compare to [5] following their protocol (sec. 4.3.3 of [5]). They evaluate on 3 parts (head, body, and legs) of the 6 animal classes of PASCAL-Part, using Percentage of Correctly estimated Parts as measure (PCP). They also need an extra measure called Percentage of Objects with Part estimated (POP), as they compute PCP only over object instances for which their system outputs a detection for that part class. Additionally, they use ground-truth object bounding-boxes at test time. More precisely, for each ground-truth box, they retain the best overlapping object detection, and evaluate part detection only within it. As table 2 shows, we outperform [5] on PCP, and our POP is substantially better, demonstrating the higher recall reached by our method. We note how [5] only report results in this easier setting, whereas we report results in a fully automatic setting without using any ground-truth at test time (sec. 5.1).

**Fine-grained [1–3].** These fine-grained recognition works report part detection results on the CUB200-2011 [17] bird dataset for the head and body. They all evaluate using PCP and including object ground-truth bounding-boxes at test time. Our model outperforms [1, 2] by a large margin and is comparable to [3]. Only [1] report results without using object ground-truth at test time. In this setting, our method performs almost as well as with object ground-truth at test time, achieving a very remarkable improvement (+25.8 PCP) compared to [1]. Furthermore, we note that CUB200-2011 is an easier

Comparison to	Chen et al. [5]	Fine-grained [1] [2] [3]	Gkioxari et al. [11]
Dataset	PASCAL-Part	CUB200-2011	PASCAL VOC09
Measure	POP	PCP	AP (0.3)
Obj GT at test	✓	✓	✓
Theirs	44.5	70.5	38.7
Ours	51.3	72.6	53.6 (65.5)

Table 2. **Comparison to other methods.** We compare to methods that report bounding-box part detection results, using their settings and measures.

dataset than PASCAL-Part, with typically just one, large, fully visible bird instance per image.

**Gkioxari et al. [11].** This action and attribute recognition work reports detection results on three person parts (head, torso, legs) on PASCAL VOC 2009 images (tab. 1 in [11]). As these do not have part ground-truth bounding-boxes, they construct them by grouping the keypoint annotations of [58] (sec. 3.2.2 of [11]). For an exact comparison, we train and test our full model using their keypoint-derived bounding-boxes and use their evaluation measure (AP at various IoU thresholds). We also report (in parenthesis) results using the standard part ground-truth bounding-boxes of PASCAL-Part during both training and testing (as PASCAL VOC 2009 is a subset of PASCAL-Part). We outperform [11] using their bounding-boxes, and obtain even better results using the standard bounding-boxes of PASCAL-Part. Moreover, we note how their part detectors have been trained with more expensive annotations (on average 4 key-points per part, instead of one bounding-box).

## 6. Conclusion

We presented a semantic part detection model that detects parts in the context of their objects. Our model includes several types of object information. We incorporate object class and appearance as indicators of what parts lie inside. We also model relative location information conditioned on the object appearance. All these complementary cues are effectively combined, achieving more accurate part detections. Our model leads to a substantially better performance than detecting parts based only on their local appearance, improving by +5 mAP on the baseline model on the



PASCAL-Part dataset. As future work we plan on jointly training the network for both object and part detection.

**Acknowledgements** This work was supported by the ERC Starting Grant VisCul.

## References

- [1] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, 2014. 1, 2, 6, 8
- [2] Di Lin, Xiaoyong Shen, Cewu Lu, and Jiaya Jia. Deep lac: Deep localization, alignment and classification for fine-grained recognition. In *CVPR*, 2015. 1, 2, 8
- [3] Han Zhang, Tao Xu, Mohamed Elhoseiny, Xiaolei Huang, Shaoting Zhang, Ahmed Elgammal, and Dimitris Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *CVPR*, 2016. 1, 2, 8
- [4] O. M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 1, 2
- [5] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. 1, 2, 3, 6, 8
- [6] Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan L Yuille. Joint object and part segmentation using deep learned potentials. In *ICCV*, pages 1573–1581, 2015. 1, 2, 6
- [7] Jiongxin Liu, Yinxiao Li, and Peter N Belhumeur. Part-pair representation for part localization. In *ECCV*, 2014. 1, 2
- [8] M. Sun and S. Savarese. Articulated part-based model for joint object detection and pose estimation. In *ICCV*, 2011. 1, 2
- [9] Norimichi Ukita. Articulated pose estimation with parts connectivity using discriminative local oriented contours. In *CVPR*, 2012. 1, 2
- [10] Wei Yang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. In *CVPR*, 2016. 1, 2
- [11] G. Gkioxari, R. Girshick, and J. Malik. Actions and attributes from wholes and parts. In *ICCV*, 2015. 1, 2, 8
- [12] Andrea Vedaldi, Siddarth Mahendran, Stavros Tsogkas, Subhrajyoti Maji, Ross Girshick, Juho Kannala, Esa Rahtu, Iasonas Kokkinos, Matthew B Blaschko, Daniel Weiss, et al. Understanding objects in detail with fine-grained attributes. In *CVPR*, 2014. 1, 2
- [13] Ning Zhang, Ronan Farrell, Forrest Iandola, and Trevor Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *ICCV*, 2013. 1, 2
- [14] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *ICCV*, 2015. 1, 2
- [15] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, 2015. 1, 2
- [16] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2, 3, 4, 5, 6, 7
- [17] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 2, 8
- [18] Ning Zhang, Manohar Paluri, Marc’Aurelio Ranzato, Trevor Darrell, and Lubomir Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *CVPR*, 2014. 2
- [19] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2
- [20] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In *ECCV*, pages 125–143, 2016. 2
- [21] Fangting Xia, Peng Wang, Liang-Chieh Chen, and Alan L Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. In *ECCV*, pages 648–663, 2016. 2
- [22] Efstratios Gavves, Basura Fernando, C.G.M. Snoek, Arnold WM Smeulders, and Tinne Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, 2013. 2
- [23] Christoph Goering, Erid Rodner, Alexander Freytag, and Joachim Denzler. Nonparametric part transfer for fine-grained recognition. In *CVPR*, 2014. 2
- [24] I. Endres, K. Shih, J. Jiaa, and D. Hoiem. Learning collections of part models for object recognition. In *CVPR*, 2013. 2
- [25] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on PAMI*, 32(9), 2010. 2, 4, 7
- [26] Yi-Hsuan Tsai, Onur C Hamsici, and Ming-Hsuan Yang. Adaptive region pooling for object detection. In *CVPR*, 2015. 2
- [27] Jianyu Wang and Alan Yuille. Semantic part segmentation using compositional model combining shape and appearance. In *CVPR*, 2015. 2
- [28] Jiongxin Liu, Angjoo Kanazawa, David Jacobs, and Peter Belhumeur. Dog breed classification using part localization. In *ECCV*, 2012. 2
- [29] Shaoli Huang, Zhe Xu, Dacheng Tao, and Ya Zhang. Part-stacked cnn for fine-grained visual categorization. In *CVPR*, 2016. 2
- [30] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *ECCV*, pages 717–732. Springer, 2016. 2
- [31] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *ICCV*, pages 3676–3684, 2015. 2

- [32] Tuan-Hung Vu, Anton Osokin, and Ivan Laptev. Context-aware cnns for person head detection. In *ICCV*, pages 2893–2901, 2015. 2
- [33] M. Juneja, A. Vedaldi, CV Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013. 2
- [34] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari. Do semantic parts emerge in convolutional neural networks? *arXiv preprint arXiv:1607.03738*, 2016. 2
- [35] N. Dalal and B. Triggs. Histogram of Oriented Gradients for human detection. In *CVPR*, 2005. 2
- [36] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2
- [37] Xianjie Chen and Alan L Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *NIPS*, pages 1736–1744, 2014. 2
- [38] H. Harzallah, F. Jurie, and C. Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009. 2
- [39] K. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the trees: A graphical model relating features, objects, and scenes. In *NIPS*, 2003. 2
- [40] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2):153–167, 2003. 2
- [41] B. Russell, A. Torralba, C. Liu, R. Ferugs, and W. Freeman. Object recognition by scene alignment. In *NIPS*, 2007. 2
- [42] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. In *CVPR*, 2011. 2
- [43] J. Yang, B. Price, S. Cohen, and M.-H. Yang. Context driven scene parsing with attention to rare classes. In *CVPR*, 2014. 2
- [44] Davide Modolo, Alexander Vezhnevets, and Vittorio Ferrari. Context forest for object class detection. In *BMVC*, 2015. 2
- [45] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2
- [46] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 2013. 3, 6
- [47] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3, 7
- [48] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 4
- [49] ZongYuan Ge, Christopher McCool, Conrad Sanderson, and Peter Corke. Subset feature learning for fine-grained category classification. In *CVPR*, pages 46–52, 2015. 4
- [50] Marcel Simon, Erik Rodner, and Joachim Denzler. Part detector discovery in deep convolutional neural networks. In *ACCV*, 2014. 5
- [51] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 5
- [52] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. on PAMI*, 2012. 6
- [53] P. Dollar and C. Zitnick. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 6
- [54] Imagenet large scale visual recognition challenge (ILSVRC). <http://www.image-net.org/challenges/LSVRC/2012/index>, 2012. 6
- [55] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010. 6, 7
- [56] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for MATLAB. In *ACM Multimedia*, 2015. 6
- [57] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 8
- [58] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *ECCV*, 2010. 8